

Messaging

Between ODrawi (OD, Ocpn_Draw_pi), Watchdog_pi (WD) and Weather_routing_pi (WR).

OpenCPn Draw and Watchdog

User aspects

First of all, we should see OD and WD as separate plugins. In OD you can draw geo-referenced objects (lines, points, straight lined areas and circles). WD knows when and how to sound alarms.

In OD, a graphical indication (crosshatching or shading) may be added to areas, to indicate whether these are intended to avoid (crosshatched inside) or to stay within (crosshatched outside), or whatever other meaning you want to give to those graphical indications.

In WD you can select whether (certain types of) alarms should react only for areas that are flagged (in OD) as to avoid or to stay within, or for all areas. On top of that, in WD you can indicate whether this should be done only for areas labeled in OD as active, or inactive, or both.

The WD Boundary Alarm has 4 different types:

1. Alarm when approaching an area from outside (based on distance);
2. Alarm when approaching an area from outside (based on time);
3. Alarm to indicate whether your boat is inside or outside an area (Inclusion Alarm, another type of anchor alarm);
4. Alarm to indicate whether AIS objects are present in an area.

For the first two alarms the WD uses the same terms for the boundary that OD does as well as allowing a check for the state of the boundary. The third alarm only looks at a specific boundary which is identified by the boundaries GUID. The fourth alarm specifies a boundary to check if an AIS target is inside it.

Beside the 4 types of Boundary Alarms mentioned above, WD has the following alarm functionality:

1. Alarm when approaching coastlines (Landfall Alarm; 2 types: time and distance)
2. Alarm when NMEA-data stream stops (NMEA Alarm)
3. Deadman Alarm
4. Alarm when distance to reference position exceeds a set value (Anchor Alarm)
5. Alarm when course over ground deviates more than set (Course Alarm; 3 types: only port deviation, only starboard deviation or a general deviation);
6. Alarms when speed deviates more then set (Speed Alarm; two types: overspeed for more than set maximum, and underspeed for less than set minimum).

In total there are 14 different types of alarms.

Technical aspects

WD and OD are independent plugins. OD knows about drawing geo-referenced objects, WD knows how to sound alarms. Now the two can work together by passing and receiving messages, in this case JSON messages (basically a text string of elements and values).

For the alarms, when WD needs boundary information, WD asks OD, via a message, whether a Lat/Lon is inside a boundary. WD can add further requirements asking for boundaries in a particular state and a particular type. Both the state and type are the same as what OD uses, i.e. Active/Inactive and Exclusion/Inclusion/Neither, or the inclusive 'Any' (meaning any type and/or any state, not being as selective).

In OD the boundaries checked are both an OD Boundary and an OD Boundary Point with range rings showing. Boundaries and Boundary Point Range Rings are both considered boundaries. The type of boundary applies to both, but the state (active/inactive) currently only applies to Boundaries, not Boundary Points. This is because there is currently no state for a Boundary Point. This may change in future updates to the plugins for consistency.

When OD completes its check of Lat/Lon inside boundaries it replies with a single message containing the first boundary that the Lat/Lon is inside AND which matches the type and state requested. The response message contains the Name, Description, GUID, Type and State of the boundary found.

WD uses the returned message to decide whether to sound the alarm and uses some of the information in the messages that are then displayed to the user, i.e. a change in text in the watchdog window and a message box, if requested.

Messaging in OCPN is synchronous, broadcast such that every plugin that registers for messages and the main program, OpenCPN, will receive every message sent. All processing of messages is synchronous, i.e. each plugin has to process each message completely and return to OCPN (the controller) before the next plugin can process the message. For the WD/OD message stream WD sends a message to OD, OD processes the message and sends a response message to WD, WD mainline processes the response message and stores the information, then returns control to OD which then returns control to WD at the point that WD created the first message. Now WD can process the saved information from OD, and the controller, OpenCPN can process the next message.

OD messages can be used by any plugin and OCPN itself to obtain information. For the OD messaging there is a "structure" for the content of the message, specifying the source requester, the type of message (Request/Response), the message i.e. FindPointInAnyBoundary, the message id (may contain an identifier for the source requester) and then the message contents, i.e. Lat, Lon, Type, etc.

So a request looks like:

```
Source: "WATCHDOG_PI"
Type: "Request"
Msg: "FindPointInAnyBoundary"
MsgId: "distance"
lat: 54.0001
lon: -30.1001
```

```
BoundaryType: "Exclusion"  
BoundaryState: "Active"
```

This message is then given a 'destination', in this case "OCPN_DRAW_PI", when the call to the OCPN messaging process is made.

The response will look like:

```
Source: "OCPN_DRAW_PI"  
Type: "Response"  
Msg: "FindPointInAnyBoundary"  
MsgId: "distance"  
GUID: "62ec7520-b58f-4087-b077-ae1c581dfec1"  
lat: 54.0001  
lon: -30.1001  
Name: "Rocks"  
Description: "Good fishing"  
Found: false  
BoundaryObjectType: "Boundary"  
BoundaryType: "Exclusion"
```

This message is then given a "destination" of the originator, in the case above "WATCHDOG_PI", when the call to the OCPN messaging process is made.

The "destination" is used so that each recipient of the broadcast message can easily check if the message is meant for it. There is no filtering provided by OCPN messaging on this value.

Using this construct there are validation checks to make sure messages are valid to process. If they are not valid there will be error messages entered into the "opencpn.log" file with relevant information.

Currently this message construct is used by OD, WD, WR (Ocpn_Draw_pi, Watchdog_pi, Weather_Routing_pi) and the AIS processing in OCPN when it broadcasts AIS information to OCPN and the plugins. In some cases there is no response message expected, i.e. AIS just sends messages, but in others the response is important.

OD is not concerned where the message came from or why, it will just respond to message requests with what is found from inspection of OD objects. WD just wants to know if it should sound an alarm or not, so it sends message requests to OD to determine certain conditions. WR just wants to know if the current Lat/Lon is valid for further processing or not, so it sends message requests to OD to determine certain conditions. AIS just provides information on each target it is dealing with.

Now the check frequency in the WD alarm screen determines how often to check for a Lat/Lon being in a boundary. One other item which should be mentioned, is that for each boundary check based on time there are up to 11 Lat/Lon messages sent to OD, for each distance check there are up to 163 Lat/Lon messages to OD. Therefore the amount of this message traffic is something to watch.

Please note that a JSON message does not have a "structure" per se, the message consists of element/value pairs written as delimited strings. The elements can occur in any order. So "structure" in

the sense used in this document really refers to required elements.

From:
<https://opencpn.org/wiki/dokuwiki/> - **OpenCPN Manuals**

Permanent link:
https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:developer_manual:developer_guide:messaging

Last update: **2019/02/28 13:53**

